Modeling, Extraction and Reuse of Organizational Knowledge

Behrouz H. Far
Faculty of Engineering, Saitama University,
255 Shimo-okubo, Urawa 338-8570, Japan

Abstract

In this paper we argue that multiagent system design can be formalized by borrowing Artificial Intelligence and Software Engineering concepts and techniques such as ontologies, organization, decomposition and synthesis. We particularly focus on ontology sharing of software agents, define agencies as organizations of agents, propose a method to conceptualize the ontology of the domain using a multi-layered bipartite graph called *Symbol Structure* (SS) and propose a method to extract organizational information from the SS of interacting agents. This information will be used in multiagent system design and development.

1 Introduction

There are already a number of projects focusing on agent-based solutions for various domains [1, 8, 10, 11]. An increasing number of projects are being revised, restructured and reconstructed in terms of software agents. Software agents are considered as a new experimental embodiment of computer programs and are being advocated as a next generation model for engineering complex, multi-platform, scalable, open, distributed and networked systems [12].

There are a number of approaches to agent design, such as: treating agents as objects, agents as expert systems, reactive agents, agents with memory and state [2], etc. However, multiagent system development is currently dominated by informal guidelines, heuristics and inspirations rather than formal principles and well-defined engineering techniques. There is no standard way of incorporating agent-oriented viewpoint into design and development of multiagent software systems.

In this paper we argue that multiagent system development can be formalized via borrowing and incorporating concepts and techniques from Software Engineering (SE) and Artificial Intelligence (AI). Specifically, we use conceptual theories such as *ontologies* and *organization*, and techniques such as *decomposition* and *synthesis*.

1.1 Ontology and Organization

Ontology in AI refers to a set of concepts or terms that can be used to describe some area of knowledge or build a representation of it. Interest in ontologies has grown due to interests in reusing or sharing knowledge across systems. Developing reusable ontologies that facilitates sharing and reuse is a goal of ontology research [5]. In multiagent system design, ontologies that can encompass agent's internal knowledge as well as the system's organizational knowledge should be developed.

Organization is a goal directed coalition of agents in which the agents are engaged in one or more tasks and knowledge and capabilities are distributed among the agents. In organization analysis a main goal is deriving organizational properties, i.e., identifying and managing relationships between the various system components.

In this work, we identify two types of organizational relationships: *signal level* and *symbol level* relationships. We argue that the signal level accounts for dynamic message passing and can be associated with *task level ontology.* At this level, messages between two communicating agents are interpreted via ascribing the same meaning to the constants used in the messages. In this way, mutual understanding of the domain constants before further message passing is guaranteed.

Symbol level relationships account for dynamic knowledge sharing and is associated with *domain ontology.* Managing organizations, i.e., organizational formation, maintenance and updating at the symbol level is a missing point in the other works.

Dynamic knowledge sharing requires knowledge level communication. Various kinds of Agent Communication Language (ACL) have been proposed. In this project, we suggest a way of systematically building the knowledge base of individual agents and blending it with the ontology of the domain, in a seamless way, using a multi-layered graph called *symbol structures* (SS). Knowledge sharing is treated as revealing the internal symbol structure to the other agents (see Sect. 2.5). In other words, two communicating agents, first share their internal symbol structure and then organizational information is derived dynamically based on identification of repetitive or persisted patterns on the profile of actions of the agent pair.

1.2 Decomposition and Synthesis

Problem decomposition is a common practice in software engineering to handle complexity. Decomposition is dividing a large problem into smaller and more manageable units each of which can then be dealt with in relative isolation.

In synthesis, one first defines a subclass of problem to be solved and builds a simplified model or prototype system that will be later incrementally updated to account for additional properties. Synthesis works exactly opposite to decomposition, which is quite common in conventional software engineering practices.

In multiagent system design, one can decompose the problem in terms of functionally distinguishable agents and then synthesis the system using those agents. Thus the whole system is simply the collection of independently developed abstract software agents that are interacting dynamically. Note that this notion of decomposition is based on functions rather than data/objects and is different from conventional reductionism approach to decomposition in conventional object-oriented SE.

The above mentioned concepts and techniques are described in detail in the following sections. In Section 2 we present the organization and a method to derive organizational knowledge. It is followed by a discussion on advantages of the proposed technique in Section 3 and a conclusion in Section 4.

2 Organization

Organizations, of various forms, physical, cognitive, temporal and institutional have been studied in operation research, management and computer sciences. The game theoretic approach to study organization focuses on modeling and suggesting computational algorithms for certain aspects of the coalition, such as social welfare [13], individual rationality, voting consensus, etc. The computational approach focuses on identifying general principles of organization and their exceptions. The proposed theories extend the information processing capabilities of individual agents to an organization level, through defining concepts such as, *bounded rationality* [14]. There is another approach focusing on environment centered analysis and design of coordination mechanisms [6][1].

The already proposed organizational models for multiagent systems have certain drawbacks. First, they cannot explain the organizational knowledge in terms of its comprising agents without reference to any other intermediary concepts. Second, they cannot provide frameworks for comparing and evaluating different organizations. Third, the organizational knowledge base cannot be updated dynamically, accounting for different configuration of the participant agents. Finally, they cannot explain the need for services of a certain agent in an organization. All of these factors are necessary in organization design and are addressed in our research.

In this project, organizational knowledge, is defined as a property of at least a pair of interacting agents. A computation method to devise organizational knowledge is proposed. This method is based on two underlying assumptions: Intelligence of Pair (IoP) and History of Patterns (HoP) described below.

2.1 Assumptions

2.1.1 Intelligence of Pair (IoP)

There is a fundamental question whether OI resides in an agent itself, external to the agent, or an outcome of the interaction among agents.

At the first glance, it seems that OI includes knowledge distribution (who knows what?) and sharing (how it can be utilized?) as mentioned in [4]. All of the proposed theories and formalisms have implicitly assumed that OI exists and implemented either inside a single agent or external to it using a meta-agent (e.g., directory and ontology service agents). However there are certain difficulties in both logical formulation and actual implementation of such theories. This is mainly due to ignoring the dynamic interactions among the agents when devising the components of OI.

In a purposeful (i.e., not random) organization, OI is a property of interaction among agents and can only

[1]For a survey on computational organization theories see Reference [3].

be ascribed to at least a pair of agents. We call this '*Intelligence of Pair (IoP)*' assumption.

2.1.2 History of Patterns (HoP)

In a biological coalition, participants may have a kind of *role* or function (during interaction with the other participants), if they show some persistence in their profile of actions over time. The same could be devised for an artificial coalition. As a matter of fact, it is not difficult to find organizations that display non-random persistent and repeated patterns of actions [4].

Agents act and perform in a physical world. Their past experiences can be recorded and explained in terms of their *histories*, that is, their *profile* of actions and *states* that they go through.

Intuitively, histories can display certain patterns. A basic feature of state representation is that it assigns a certain characteristic to its reference agent. Therefore it is possible to define OI patterns with reference to agents' history.

We argue that OI patterns emerge from discovering a persisted state or an ordered pattern in the agent's profile of actions. We call this '*History of Patterns (HoP)*' assumption.

In a biological coalition, persistence is considered to be the most interesting characteristic and is believed to be governed by natural selection law. In an artificial coalition, besides persistence, other kinds of ordered patterns, such as repetition cycles, may also be considered as important features of the coalition.

IoP and HoP assumptions account for dynamic interactions and a computation method based on this assumption is proposed in Sect. 2.6.

2.2 Modeling

The domain ontology is represented by *Symbol Structure* (SS). The SS is a finite connected multi-layer bipartite graph. There are two kinds of nodes in each layer of SS: concepts (c) and relations (r) (See Fig. 1). One source of difficulty when processing concepts, is distinguishing a concept at various levels of abstraction, as well as differentiating between generic concepts and their instances. Function type is defined to ease such differentiation. The function type maps concepts and relations onto a set T. The elements of T are called type labels. Type hierarchy provides a means of evaluating a concept at various levels. The type hierarchy is a partial ordering defined over the set of type labels, T. In this way, moving from any level of abstraction to another is supported.

Agents take SS as their private knowledge model. They possess a common symbol set and a number of actions that operate upon the SS. Knowing the common symbol set and actions, together with the assumption that the problem solving behavior is concerned with the representations by certain general principles, makes it possible to explain an important segment of the regularities in problem solving behavior of those agents.

Flexibility, extendibility and interoperability are three main advantages of SS. We can mention that SS is semantically richer than semantic networks because both the concepts and relations are augmented with types. SS can be further used to develop rule-based or network-based knowledge bases for individual agents. Furthermore, in this formalism, as the actions take concepts and relations as their attributes, flexibility in representing concepts in SS, will increase the probability to achieve a goal.

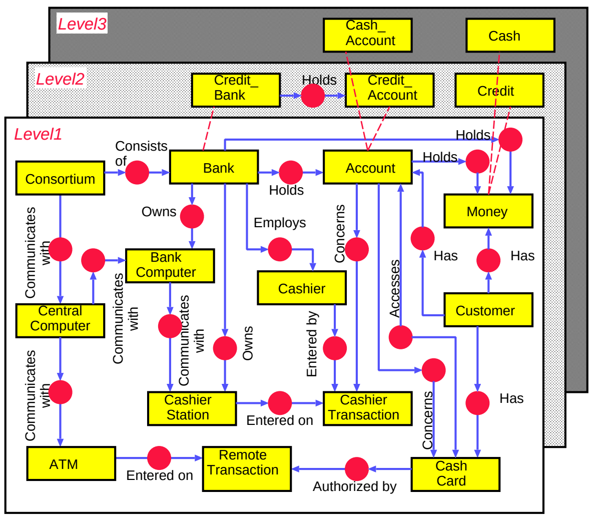

Figure 1: Example of a triple-layer SS

2.3 Manipulating Symbol Structures

SS is powerful enough to represent concepts and relationships at various levels of abstraction. In this section we show how to manipulate symbol structures.

Basically, there are *combination* and *insight* rules. Combination rules are the followings:

Join rule: Join rule merges identical concepts. If a concept c in u is identical to a concept d in v, then let w be the symbol structure obtained by deleting d and linking to c all arcs of relations that had been linked to d.

Simplification rule: Redundant relations of the same type linked to the same concept in the same order can be reduced by deletion all but one. If the relations r and s in the symbol structure u are duplicates, then one of them may be deleted from u together with all its arcs.

An *insight* is defined as the ability to *generalize* and/or *specialize* information from a symbol structure. Specialization is equivalent to increasing the precision in modeling. The outcome of applying specialization rules on a SS is another SS which is more restricted than the original one. Conversely, generalization proceeds in the reverse order of specialization.

Generalization / Specialization rule : For two arbitrary levels u and v of any symbol structure, if u is identical to v except that some type labels of the nodes of v are restricted to subtypes of the same nodes in u, then u is called a specialization of v, written $u \prec v$, and v is called a generalization of u.

Generalization defines a partial ordering among the levels of any symbol structure, called the generalization hierarchy. Specialization is equivalent to replacing the type label of a concept with the label of a subtype.

Note that specialization rule is not a rule of causal inference. It only enforces selectional constraints by preventing certain combinations from being derived. Obviously, opposite to specialization, generalization preserves truth but does not necessarily preserve selectional constraints.

An important feature of SS is that, two different symbol structures may have a common generalization (and/or specialization).

2.4 Actions and States

There are finite sets of *actions* and *states* associated with each agent. Given a set of all possible actions, A, an agent's action is a subset $B \subseteq A$.

Both *actions* and *states* take concepts and relations as their attributes. For example, moving money from one account, say x, to the other, say y, is associated with an action `MOVE`, where,

$$\texttt{MOVE}(\mathit{Money}, \mathit{Account}\texttt{:-x}, \mathit{Account}\texttt{:-y})$$

means that action `MOVE` works on three concepts, a generic type *Money* and two instances *Account* : `-x` and *Account*`:-y`.

Similarly, the *state* of the world after performing this action is

$$\texttt{S}_1(\mathit{Money}, \mathit{Account}\texttt{:-y})$$

2.5 Interaction Among Agents

Ontology sharing by moving from one agent to another and on an organizational basis requires defining the basic agent interactions, i.e, *cooperation*, *coordination* and *competition.* For a pair of agents to interact, each should maintain a model of the other agent, as well as a model of future interactions [9].

Cooperation: Cooperation is revealing an agent's *goal* and the knowledge behind it, i.e., its symbol structure to the other party. In cooperation both agents have a common goals.

Coordination: Coordination is revealing an agent's *goals* and the knowledge behind it, i.e., its symbol structure to the other party. In coordination, agents have separate goals.

Loose Competition: Loose competition is revealing only an agent's *goals* but masking the knowledge behind it to the other party.

Strict Competition: Strict competition is neither revealing an agent's *goals* nor the knowledge behind it to the other party.

Knowledge sharing is equivalent to merging two or more symbol structures using *combination* and *insight* rules. Knowledge sharing is fully utilized in the case of *cooperation* and *coordination.* In case of loose or strict competition the SS model of the other agent should be predicted based on its observable states. This is a topic yet to be investigated.

2.6 Deriving Organizational Information

Here we propose a computation method for generating OI concepts based on the IoP and HoP assumptions (see Sect. 2) using Symbol Structure. Fig. 2 depicts the idea. In this method, first, a pair of agents are selected and by using manipulation rules (see Sect. 2.3) their *pairwise profile* is produced (see below). Then by using a simple pattern detection algorithm, possible repetition and persistence patterns are derived and added to the organizational knowledge base.

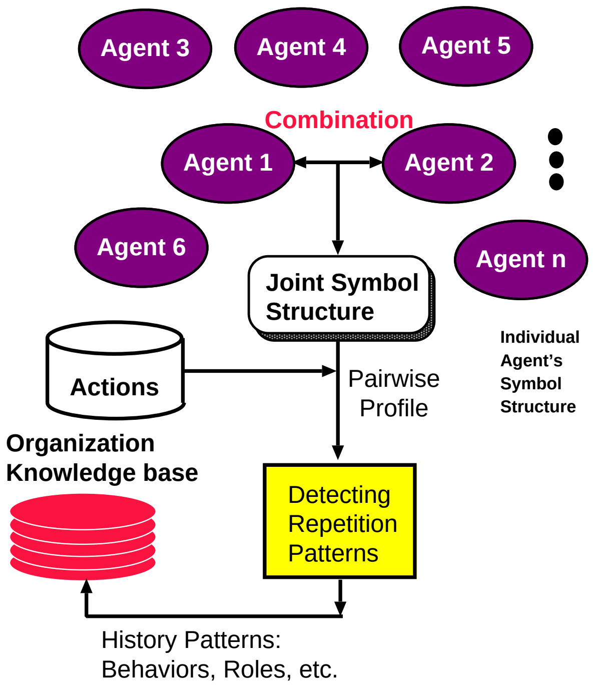
Agent 3
Agent 4
Agent 5
Combination
Agent 1
Agent 2
Agent 6
Agent n
Joint Symbol Structure
Actions
Pairwise Profile
Individual Agent's Symbol Structure
Organization Knowledge base
Detecting Repetition Patterns
History Patterns: Behaviors, Roles, etc.

Figure 2: Organization Intelligence concept

An agent's profile, $\mathcal{P}$ is a finite sequence of (s, a), recorded during reasoning.

Where, s is problem solving state; a is action; $a \in B$; $B \subseteq A$; B is the action set available to agent and A is the set of all possible actions.

A pairwise (i.e., joint) profile of two agents, i and j, $\mathcal{P}_{ij}$ is a finite sequence of $([s_i, s_j], [a_i, a_j])$, during reasoning, after merging their symbol structures. Each action is a pair of joint actions of the two agents.

$$\forall [a_i, a_j] \quad a_i \in B_i, \quad a_j \in B_j$$

$B_i \subseteq A$ and $B_j \subseteq A$ are the action set available to agent i and j, respectively.

3 Advantages

Although there are many projects focusing on multiagent system design and development, there is no project aiming at dynamic acquisition of organizational knowledge and reuse of this knowledge in design of multiagent system based on dynamical coalition. Other distinctive features of this project are:

1) **Providing solution to the infamous knowledge sharing problem**
In this project, we suggest a way of systematically building the knowledge base of individual agents and blending it with the ontology of the domain, in a seamless way, using Symbol Structures. Knowledge sharing is treated as revealing the internal Symbol Structure to the other agents. In other words, two communicating agents, first share their internal Symbol Structure and then messages are interpreted. Interpretation is ascribing the same meaning to the constants used in the message. In this way, mutual understanding of the domain constants before further message passing is guaranteed.

2) **Giving a unified view of the agent problem solving issues**
There are already a number of approaches to distributed problem solving, using agent technology, such as blackboard systems, broadcast methods, delegation, contract networks, etc. A main goal in multiagent problem solving is to maintain the coherent performance of the agents while maintaining their autonomy. Cooperation, coordination and competition are three major problems is multiagent problem solving. A number of protocols for cooperation, coordination and competition have been suggested (for a survey see [9]). However, there is no single formalism that can cover them all. The technique proposed in this paper gives a coherent view of cooperation, coordination and competition in a distributed problem solving environment.

3) **Advantages of deriving organizational knowledge based on IoP and HoP assumptions**
Deriving organizational knowledge based on IoP and HoP assumptions can fully cover the problems with the already proposed organizational models mentioned in Section 2.

4 Conclusions

In this paper, we defined the organizational knowledge and devised a computational method to extract it, using ontology of the domain and public knowledge of interacting agents. This is quite useful for designing agent coalition which is shaped dynamically and defining multiple roles for individual agents when participating in a number of different coalitions.

A distinguishing point was attributing the organizational knowledge to at least a pair of agents. The knowledge of an agent is represented by symbol structure and agents can share their knowledge using combination, specialization and generalization rules. We gave a coherent view of agent interaction, i.e., cooperation, coordination and competition problems. Dealing with uncertainty when predicting the SS model of the other agents in case of loose or strict competition and blending SS with the belief networks is a future research topic.

Applications using the framework and techniques described in this paper, such as a multiagent system for electronic commerce [7], a multiagent intelligent tutoring system (ITS) system, a computer aided software engineering (CASE) tool for object oriented software design are under investigation and development.

References

[1] J.M. Bradshaw, ed., *Software Agents,* 480 p., MIT Press, 1997.

[2] W. Brenner, R. Zarnekow and H. Wittig, *Intelligent Software Agents,* Springer Verlag, Berlin, 1998.

[3] K.M. Carley and M.J. Prietula, *Computational Organization Theory,* Laurence Erlbaum Associates, Hillsdale, NJ, 1994.

[4] K.M. Carley and L. Gasser, "Computational Organization Theory," in Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, ed. G. Weiss, G., pp.299-330, MIT Press, 1999.

[5] B. Chandrasekaran, J.R. Josephson and V.R. Benjamins, "What Are Ontologies, and Why Do We Need Them?" IEEE Intelligent Systems & Their Applications, vol.14, no.1, pp.20-26, 1999.

[6] B. Decker, "TAEMS: A Framework for Environment Centered Analysis and Design of Coordination Mechanisms," in Foundations of Distributed AI, eds. G.M.P. O'Hare and N.R. Jennings, John Wiley & Sons, 1996.

[7] B.H. Far, S.O. Soueina, H. Hajji, S. Saniepour and A.H. Hashimoto, "An Integrated Reasoning and Learning Environment for WWW Based Software Agents for Electronic Commerce," *Transactions of IEICE*, Vol. E81-D No. 12, pp. 1374-1386, 1998.

[8] M.N. Huhns and M.P. Singh, *Readings in Agents,* Morgan Kaufmann Publishers, 1997.

[9] M.N. Huhns, L.A. Stephens, "Multiagent Systems and Societies of Agents," in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, Weiss, G., edt., pp. 79-120, MIT Press, 1999.

[10] N.R. Jennings, K. Sycara and M. Wooldridge, "A Roadmap of Agent Research and Development," Autonomous Agents & Multiagent Systems, vol.1, pp.7-38, 1998.

[11] N.R. Jennings and M. Wooldridge, eds., *Agent Technology, Foundations, Applications and Markets,* Springer-Verlag, 1998.

[12] N.R. Jennings, "On agent-based software engineering," Artificial Intelligence, vol. 117, pp. 277-296, 2000.

[13] T.W. Sandholm, "Distributed Rational Decision Making," in Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, ed. G. Weiss, pp.201-258, MIT Press, 1999.

[14] H.A. Simon, *Administrative Behavior,* Free Press, New York, 1947.